TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

PROGRAMOVÁNÍ SLUŽEB PRO PŘENOS HLASU PO IP SÍTÍCH NA PORTÁLECH PRO SDÍLENÝ VÝVOJ APLIKACÍ

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Martin Bureš**
Vedoucí práce: Ing. Jana Vitvarová, Ph.D.

Liberec 2014

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

PROGRAMMING SERVICES FOR VOICE OVER IP NETWORKS USING WEB-BASED HOSTING FOR SOFTWARE DEVELOPMENT

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Martin Bureš**
Supervisor: Ing. Jana Vitvarová, Ph.D.

Liberec 2014

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin Bureš**
Osobní číslo:	**M11000066**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Programování služeb pro přenos hlasu po IP sítích na portálech pro sdílený vývoj aplikací**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Proveďte rešerši open source projektů nad VoIP ústřednou Asterisk na hostingových portálech pro sdílený vývoj aplikací se správou verzí jako je Github, nebo Bitbucket.
2. Nastudujte si a popište způsob práce na takovýchto portálech a vyzkoušejte si přispět do vybraného projektu.
3. Na jednom z portálů si založte vlastní projekt (privátní nebo veřejný).
4. V rámci tohoto projektu rozšiřte referenční VoIP řešení nad ústřednou Asterisk z ročníkové práce o podporu IVR (Interactive Voice Response) a vytvořte webové ovládací rozhraní pro tuto ústřednu, které bude poskytovat následující moduly:
   a) nastavení ústředny,
   b) správu uživatelů, telefonních plánů a hlasových schránek,
   c) přehled provedených hovorů,
   d) vytváření IVR.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Hostingová služba pro tvorbu sdílených aplikací se správou verzí Github: https://github.com/**

[2] **Hostingová služba pro tvorbu sdílených aplikací se správou verzí Bitbucket: https://bitbucket.org/**

[3] **Mahler P.: VoIP Telephony with Asterisk, ISBN 09759992-0-6**

[4] **Referenční průvodce pro VoIP dostupný na: http://www.voip-info.org/wiki/**

[5] **Dokumentace k PHP dostupná na: http://php.net/**

[6] **Dokumentace k MySQL dostupná na http://dev.mysql.com/doc/**

[7] **Connolly T., Begg C., Strachan A.: Database systems, Addison-Wesley, 1999**

Vedoucí bakalářské práce: **Ing. Jana Vitvarová, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2013**

Termín odevzdání bakalářské práce: **16. května 2014**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

Technická univerzita v Liberci - 14 -

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2013

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 12.5.2014

Podpis:

Poděkování

Děkuji Ing. Janě Vitvarové, Ph.D. za vedení mé bakalářské práce a za poskytnutí tohoto tématu, které mi pomohlo rozšířit své poznatky v této oblasti.

Abstrakt

Bakalářská práce se zabývá tvorbou webového ovládacího rozhraní pro telefonní ústřednu Asterisk s využitím prostředků pro sdílený vývoj aplikací. Na počátku práce jsou použité prostředky komunitního vývoje analyzovány. Další část se zabývá popisem použitých technologií nutných pro správnou funkčnost celého řešení, kterými jsou například skriptovací jazyk PHP, databázový server MySQL a samozřejmě software pro provoz telefonní ústředny – Asterisk. V praktické části se tato práce věnuje návrhu a vytvoření příslušného webového rozhraní. Nedílnou součástí praktické části je i úprava nastavení Asterisku, aby byly jednotlivé komponenty provázané a aby bylo celé řešení funkční.

Klíčová slova: VoIP, Asterisk, PHP, ovládací rozhraní, komunitní vývoj

Abstract

The bachelor thesis deals with the creation of web control interface for Asterisk PBX with resources for shared application development. Used tools for community development are analyzed at the beginning of this work. Another section deals with the description of the technologies necessary for the proper functioning of the entire solution, which is a scripting language such as PHP, MySQL database server, and of course software to operate PBX – Asterisk. Practical part of this thesis deals with the design and creation of the web interface. An integral part of the practical part is a configuration of the Asterisk to provide a functional solution with PBX and created interface.

Keywords: VoIP, Asterisk, PHP, control interface, community development

Obsah

Seznam symbolů, zkratek a termínů

IP – protokol pracující na síťové vrstvě používaný v počítačových sítích

VoIP – technologie umožňující přenos digitalizovaného hlasu v těle UDP/TCP paketů

SIP – signalizační protokol aplikační vrstvy, který slouží na vytvoření, modifikování a ukončení spojení

PBX – pobočková telefonní ústředna

CDR – systém pro ukládání informací o provedeném hovoru (z angl. Call Detail Record)

AMI – rozhraní pro správu ústředny (z angl. Asterisk Manager Interface)

HTTP – aplikační protokol určený pro komunikaci prohlížeče a webového serveru

HTML – značkovací jazyk umožňující publikaci dokumentů na internetu

XSS – typ zranitelnosti internetové stránky prostřednictvím nevhodně ošetřených vstupů (z angl. Cross Site Scripting)

SQL Injection – typ zranitelnosti internetové stránky prostřednictvím nevhodně ošetřených vstupů

CSRF – typ zranitelnosti internetové stránky při kterém je vytvořen požadavek na vykonání určité akce od nelegitimního zdroje (z angl. Cross Site Request Forgery)

IVR – interaktivní hlasová odezva, která slouží pro základní automatizovanou komunikaci s volajícím

1 Úvod

Dnešní doba dává lidstvu spoustu možností, jak navzájem komunikovat. Ať už se jedná o dnes již ustupující psaní dopisů nebo naopak moderní prostředky typu mobilní telefon nebo počítač. A právě v těchto pokročilých zařízeních jsou dnes k dispozici různé programy a způsoby pro vzájemnou hlasovou komunikaci. Společným znakem většiny těchto programů je přenos dat, např. hlasové informace, po IP sítích. Tento způsob přenosu dat se prosazuje zejména díky nižším až žádným nákladům na hovory mezi jednotlivými účastníky a také snadnou implementovatelností a ovladatelností, zejména pro účastníky.

Tato bakalářská práce navazuje na referenční ročníkovou práci [1] z roku 2013, která se zabývala konfigurací telefonní ústředny a prakticky rozšiřuje možnosti konfigurace a ovládání této ústředny pomocí webové aplikace. Způsob konfigurace přes webové rozhraní je v dnešní době hojně rozšířený, praktický a mezi jeho nesporné výhody patří přenositelnost, resp. použitelnost na více platformách a pro správnou funkci vyžaduje pouze standardní internetový prohlížeč na koncovém zařízení uživatele.

Prvním cílem této práce je seznámení s vývojem na portálech pro sdílený vývoj aplikací. A to na základě skutečnosti stejného způsobu vývoje telefonní ústředny Asterisk. Druhým cílem je vytvoření webového rozhraní pro ovládání zmíněné ústředny. Toto rozhraní bude vytvořeno jako nadstavba pro použití s ústřednou a to za pomoci vybraného portálu pro sdílený vývoj aplikací.

Jako počáteční předpoklad uvažujme, že máme k dispozici nastavenou a funkční telefonní ústřednu, která je přístupná z vnitřní i vnější sítě. Na této ústředně se dále nachází webový server určený pro provoz webové aplikace pro správu ústředny, která je hlavní náplní této práce, a také databázový server, ve kterém je uloženo nastavení telefonní ústředny. Tato zmíněná konfigurace byla náplní uvedené ročníkové práce, a proto je nadále považována za plně použitelnou pro potřeby této bakalářské práce.

Nejprve bude třeba modifikovat nastavení telefonní ústředny a přidat podporu pro IVR. Pro ukázku bude vytvořeno IVR s předpovědí počasí. Dalším cílem bude vytvořit webovou aplikaci pro správu ústředny. Do této aplikace bude mít přístup správce ústředny a běžní uživatelé. Tito běžní uživatelé budou mít k dispozici agendy správy

vlastních telefonních linek, přehledy hovorů, hlasové schránky a telefonní seznam. Správce ústředny bude moci vykonávat úkony jako běžný uživatel a navíc bude mít k dispozici správu telefonního plánu, veřejných linek a běžných uživatelů. Oba zmíněné typy účtů by také měly umožňovat správu přihlašovacích údajů a historii událostí konkrétního účtu. Poslední součástí aplikace by měla být možnost obnovit zapomenuté heslo pro oba typy účtů.

Práce je rozdělena do několika kapitol. Ve druhé kapitole jsou popsány způsoby sdíleného vývoje aplikací a použité nástroje pro takový vývoj. Rovněž tato kapitola obsahuje i rešerši významných existujících programů na těchto portálech s tematikou telefonní ústředny Asterisk. Další kapitola se zabývá použitými technologiemi v této práci. Ve čtvrté kapitole je popsána upravená konfigurace ústředny Asterisk, která je potřebná pro tuto práci. V páté kapitole je pak detailně popsána vytvořená aplikace pro ovládání ústředny.

2 Sdílený vývoj aplikací

Při vytváření projektů, zvláště pak při vývoji softwaru je, kromě samotného vyvíjeného algoritmu či obecněji tvořeného kódu, potřeba nějakým způsobem ukládat aktuální verze rozpracované práce. V lepším případě i s uživatelským komentářem k aktuálnímu stavu a možností vrátit se při vývoji do určitého bodu zpět. Je možné, že pro jednotlivce, který si vyrábí software pro vlastní použití, není takto sofistikovaný způsob správy verzí jeho projektu plně využitelný. Ale v okamžiku, kdy se vyskytne požadavek na sdílení zdrojových kódů nebo dokonce na vývoj ve větším počtu vývojářů, vyvstávají oprávněné otázky, jak spravovat jednotlivé verze vyvinutého kódu a jak tento kód mezi vývojáře distribuovat. Právě těmito otázkami se zabývá tato kapitola. Jsou v ní popsány základní systémy správy verzí i pokročilejší moderní nástroje pro komunitní vývoj. Dále se tato kapitola zabývá analýzou několika projektů vyvíjených komunitou uživatelů s tematickým zaměřením na ovládání telefonní ústředny Asterisk.

2.1 Verzování

Verzování (neboli správa verzí) je systém, který zaznamenává změny souborů v průběhu času. Uživatel tak může kdykoli v případě potřeby obnovit jejich konkrétní verzi. V některé literatuře lze nalézt i označení verzování jako systém správy verzí VCS (z angl. Version Control System). VCS umožňuje vrátit jednotlivé soubory nebo celý projekt do předchozího stavu, porovnávat změny provedené v průběhu času, zjistit autora poslední úpravy kódu, efektivně sdílet kód mezi vývojáři a mnoho dalšího. V dnešní době jsou navíc vytvořeny velice praktické nástroje pro práci s takovýmito systémy. Dávno je pryč doba, kdy bylo možné tyto systémy ovládat pouze z příkazové řádky. Samozřejmě tato možnost zůstala zachována, ale prakticky pro každý dále uvedený systém správy verzí dnes existuje uživatelská aplikace, ve které je sdílení kódu otázkou kliknutí na příslušné tlačítko a svým způsobem může vývojářům práci velmi usnadnit.

U jednotlivých vývojářů, popřípadě i v malých vývojových týmech, je možné se setkat s praxí, kdy správa verzí je prováděna prostřednictvím kopírování souborů do jiného adresáře. Takový přístup je v těchto situacích častý, protože je jednoduchý. Je s ním

však spojeno také velké riziko omylů a chyb, protože se může stát, že vývojář zapomene, ve kterém adresáři se právě nachází, a nedopatřením může přepsat stávající soubory. Právě z uvedeného důvodu byl jako jeden z prvních verzovacích systémů vyvinut systém RCS. Tento systém pracuje na lokálním principu s jednoduchou databází, která uchovává změny souborů mezi jednotlivými verzemi. Systém později může díky porovnání těchto změn vrátit jakýkoli soubor do podoby, v níž byl v libovolném okamžiku.

V okamžiku, kdy se vyskytne potřeba distribuovat vyvíjený kód mezi další vývojáře, nastane s lokálním způsobem správy verzí logický problém, protože by najednou existovalo více lokálních verzí, které by bylo obtížné synchronizovat. Řešením tohoto problému jsou tzv. centralizované systémy správy verzí CVCS (z angl. Centralized Version Control System). Tyto systémy obsahují serverovou část, která uchovává všechny verzované soubory. Z této serverové části si potom jednotliví klienti stahují soubory v příslušných verzích. Tento koncept byl dlouhá léta standardem pro správu verzí, protože nabízí mnoho výhod, zejména v porovnání s lokálními systémy. Správa centrálního CVCS je jednodušší, protože toto úložiště existuje pouze jednou a není tím pádem potřeba složitě synchronizovat lokální databáze. Další výhodou je i možnost každému vývojáři přiřadit určitý stupeň oprávnění, co smí v CVCS modifikovat. Ostatní uživatelé pak mohou sledovat jednotlivé verze pomocí historie úprav. Mezi hlavní nedostatky tohoto přístupu patří situace, když vypadne centrální úložiště, resp. CVCS. V této situaci pak není možné sdílet vyvíjený kód dále a musí se čekat do obnovení jeho činnosti. Práce jako taková teoreticky může pokračovat, ale provedené úpravy je možno do CVCS odeslat až po obnovení jeho činnosti. Tím se hodně snižuje možnost pracovat v této situaci. Další nepříjemná situace může nastat v okamžiku, kdy dojde k fyzickému poškození centrálního úložiště (např. poruše pevného disku) a data z tohoto úložiště by nebyla zálohována. V takovém případě hrozí nevratná ztráta uložených dat. Mezi zástupce CVCS lze řadit CVS a Subversion.

Právě na základě vlastností a nevýhod předešlých dvou postupů verzování vznikl třetí, dnes běžně používaný, který je nazýván jako distribuovaný systém správy verzí DVCS (z angl. Distributed Version Control System). Jako zástupce této kategorie verzovacích systémů patří Git, Mercurial a Bazaar. V systémech DVCS uživatelé pouze nestahují nejnovější verzi souborů, ale uchovávají si kompletní kopii repositáře. Pokud v takové

situaci dojde k výpadku serveru, lze jej obnovit zkopírováním repositáře od libovolného uživatele, protože každá lokální kopie u jednotlivých uživatelů je plnohodnotnou zálohou všech dat. [2]

2.2 Verzovací systémy

Z předchozí kapitoly je zřejmé, že existuje více typů systémů správy verzí. Z tohoto důvodu jsou v této kapitole jednotlivé systémy stručně popsány.

2.2.1 RCS

Systém pro správu verzí RCS (z angl. Revision Control System) vyvinul v roce 1982 Walter F. Tichy na univerzitě Purdue. V počátečních verzích uměl spravovat verze pouze textových souborů a omezeně i binárních. Tyto verze ukládal s využitím nástroje diff, který zjišťuje rozdíly mezi dvěma jednotlivými soubory. Nevýhodou bylo, že RCS neuměl zpracovávat celé verze projektů, ale pouze jednotlivé soubory. Další nevýhodou bylo, že byl od začátku navržen jako lokální systém správy verzí. Z předchozí kapitoly je tudíž zřejmé, že tento systém nebyl vhodný pro sdílení a distribuci jednotlivých verzí více uživatelům. I přes jeho stáří je stále částečně udržován. Poslední větší úpravy v jeho kódu byly dle [3] provedeny v listopadu 2013 a je licencován pod licencí GNU GPLv3.

2.2.2 CVS

CVS (z angl. Concurrent Version System) je systém souběžných verzí, který slouží ke správě verzí projektu. Systém spravuje jednu nebo několik skupin souborů nazývaných repositář. Každý repositář má vlastní řízení přístupu a je dělen na menší části, které mohou reprezentovat projekty nebo skupiny projektů ve stromové struktuře. Repositář je pak uložen ve formě souborů na serveru. Změny jsou sledovány na úrovni verzí jednotlivých souborů v projektu. Tento systém také podporuje vytváření nových vývojových větví projektu a pochopitelně i jejich sloučení. Nová vývojová větev se běžně označuje jako branch. Po vytvoření této oddělené větve jsou obě větve vyvíjeny paralelně. V případě potřeby mohou být zpět sloučeny do jedné. Pro CVS je typické použití jako víceuživatelská klient/server aplikace. Mezi nevýhody tohoto systému patří

nemožnost záznamu přesunu souboru, nemožnost verzování symbolických odkazů, nedostatečná podpora UTF kódování, problematické větvení projektu a také špatná podpora binárních souborů. Systém CVS je vyvíjen pod licencí GPL. [4]

2.2.3 Apache Subversion

Apache Subversion, dříve Subversion (SVN), je systém pro správu verzí zdrojových kódů, který byl vytvořen jako náhrada za CVS. Oficiálně vznikl 31. 8. 2001, ale první verze byla vydána až v roce 2004. Od té doby je dále vyvíjen. Dne 7. 11. 2009 byl vložen do Apache inkubátoru, což je projekt, který sdružuje začínající projekty s cílem začlenit je jako plnohodnotné do ASF (z angl. Apache Software Foundation). Nyní je vyvíjen pod licencí Apache Licence 2.0. Stejně jako CVS, je i Subversion založen na principu centrálního repositáře. K centrálnímu repositáři je obvykle možné přistoupit přes nativní protokol SVN nebo DAV (z angl. Distributed Authoring and Versioning). K přístupu lze použít i některé grafické nástroje. Samozřejmostí také zůstává možnost ovládání z příkazového řádku.

Subversion rozlišuje několik základních termínů. Repositář, neboli centrální úložiště, ve kterém se organizuje projekt a spravují jeho verze. Fyzicky je repositář uložen v souborovém systému serveru. V každém projektu může existovat více větví (tzv. branch). Ty slouží k organizaci repositáře. Pokud se z repositáře vyzvedne větev, na klientovi vznikne adresářová struktura, která přesně odpovídá větvím v repositáři. Každá změna v repositáři je označena jako revize. Ta slouží ke sledování změn ve větvích v čase. Každá změna v nějaké větvi vytvoří novou revizi v rámci celého repositáře. Revize obsahuje informace o tom, co bylo změněno, kdo změnu provedl, uživatelský komentář a čas. Při stažení obsahu repositáře se lokálně vytvoří tzv. pracovní kopie. Tato kopie dat z určité větve v aktuální revizi se uloží na pevný disk lokálního klienta. Do pracovní kopie je možné provádět změny, které je možné uložit zpět do repositáře.

Dalším termínem je tzv. odeslání do repositáře (z angl. commit). Pokud se provádí commit celé pracovní kopie, jedná se o atomickou operaci. Mezi další používané termíny patří konflikt. To je stav, který signalizuje, že stejný objekt, který má být právě odeslán, byl změněn někým jiným a nachází se v repositáři v aktuální revizi v jiné

podobě, než jaký je v pracovní kopii. Pokud se v lokální kopii objeví konflikt, je potřeba takovýto konflikt nejprve vyřešit a až poté provést odeslání. Subversion ukládá vždy jen informace o provedených změnách. Pro sloučení změn z větve v repositáři do pracovní kopie se používá termín merge. Posledním termínem je tzv. cheap-copy, která se používá k vytvoření kopie v repositáři.

Pro práci se systémem SVN uživatel nejprve provede vyzvednutí projektu (tzv. checkout) z repositáře do lokálního adresáře. Tím se vytvoří pracovní kopie. V této pracovní kopii uživatel provádí požadované změny. Po provedení všech změn (úpravy, přidání, smazání) provede uživatel odeslání do centrálního repositáře (tzv. commit). Provedené změny se zkopírují do centrálního repositáře, odkud si je další uživatelé mohou stáhnout a pokračovat v práci dále. [5]

2.2.4 Git

Git je distribuovaný systém správy verzí (DVCS) vytvořený Linusem Torvaldsem, původně pro vývoj jádra Linuxu. V letech 1991 – 2002 bylo jádro Linuxu spravováno formou záplat a archivních souborů. V roce 2002 začal projekt vývoje linuxového jádra využívat komerční systém DVCS s názvem BitKeeper. V roce 2005 se ale zhoršily vztahy mezi komunitou, která vyvíjela jádro Linuxu, a komerční společností, která vyvinula BitKeeper, a společnost přestala tento systém poskytovat zdarma. To přimělo Linuse Torvaldse, aby vyvinul vlastní nástroj, založený na poznatcích získaných při používání programu BitKeeper. Existuje implementace JGit v Javě a podpora ve formě pluginu EGit pro vývojové prostředí Eclipse. V dnešní době je Git velmi populární systém správy verzí a je používán mnoha významnými projekty např. Ruby on Rails, X.Org, atd. Git je v současné době šířen pod licencí GPLv2.

Hlavním rozdílem mezi systémem Git a ostatními systémy pro správu verzí (např. Subversion) je způsob, jakým Git zpracovává data. Většina ostatních systémů ukládá informace jako seznamy změn jednotlivých souborů. Tyto systémy chápou uložené informace jako sadu souborů a seznamů změn těchto souborů v čase. Git zpracovává data jinak. Chápe je jako sadu snímků vlastního systému souborů. Při každém uložení stavu souborů udělá Git snímky všech vašich souborů a uloží reference na tyto snímky. Pokud v souborech nebyly provedeny žádné změny, Git neukládá znovu

celý soubor, ale pouze odkaz na předchozí identický snímek, který už byl uložen. Toto je důležitý rozdíl mezi systémem Git a ostatními systémy.

Uživatel si nejprve naklonuje vzdálený repositář (tzv. clone) do lokálního repositáře. Tento lokální repositář je kopií centrálního repositáře, která má ale k dispozici i celou historii úprav. V lokálním repositáři pak uživatel provádí požadované změny. Po provedení všech změn provede uživatel označení aktuální verze (tzv. commit) a poté odeslání do centrálního repositáře (tzv. push). Provedené změny se zkopírují do centrálního repositáře, odkud si je další uživatelé mohou stáhnout a pokračovat v práci dále. Pokud již je lokální repositář vytvořen pomocí klonování vzdáleného repositáře, před každým započetím úpravy kódu, je nutné si z centrálního repositáře stáhnout aktuální verzi (tzv. pull). Právě díky tomuto postupu je Git rychlý oproti Subversion. Při práci s ním není potřeba neustálé síťové připojení a prohlížení jednotlivých revizí je záležitostí pouze na lokálním počítači.

Při ukládání změn souborů si Git vždy vytvoří SHA-1 otisk ukládaného souboru. Pokud takto vygenerovaný otisk již existuje v interní databázi Gitu, soubor se považuje za nezměněný. To je dáno vlastním výpočtem otisku, jelikož princip otisku spočívá v tom, že se generuje unikátní otisk pro unikátní posloupnost bytů, resp. souboru, a i malá změna posloupnosti bytů má za následek změnu otisku. Interně Git nevyužívá názvy souborů, ale každý soubor si identifikuje právě pomocí jeho vypočítaného otisku. [2]

2.2.5 Mercurial

Mercurial je multiplatformní verzovací nástroj a řadí se do kategorie DVCS. Je napsán v jazycích Python a C. Je určen primárně pro použití v příkazovém řádku, dostupná jsou ale i grafická uživatelská rozhraní. Poprvé ho jeho autor Matt Mackall představil 19. 4. 2005. Podnětem k jeho vytvoření bylo ukončení šíření bezplatné verze programu BitKeeper, který se používal pro správu verzí jádra Linuxu. Vývoj Mercurialu byl zahájen několik dnů po zahájení projektu Git, iniciovaného Linusem Torvaldsem se stejným cílem. Hlavními cíli vývoje byla vysoká výkonnost a škálovatelnost, decentralizace a distribuovaný vývoj, odolná správa textových i binárních souborů a možnost pokročilého větvení. Všechny tyto body měl Mercurial dosáhnout při zachování celkové jednoduchosti. Je vydán pod licencí GNU GPLv2.

Vzhledem k tomu, že Mercurial je zástupce kategorie systémů správy verzí DVCS, existuje zde opět centrální server, přes který jsou jednotlivé repositáře distribuovány. Princip fungování je velmi podobný např. Gitu. Uživatel si nejprve naklonuje vzdálený repositář (tzv. clone) do lokálního repositáře. Tento lokální repositář je kopií centrálního repositáře, která má ale k dispozici i celou historii úprav. V lokálním repositáři pak uživatel provádí požadované změny. Po provedení všech změn uživatel označení aktuální verzi a poté ji odešle do centrálního repositáře (tzv. push). Pokud již je lokální repositář vytvořen pomocí klonování vzdáleného repositáře, před každým započetím úpravy kódu je nutné si z centrálního repositáře stáhnout aktuální verzi (tzv. pull). [6]

2.2.6 GNU Bazaar

GNU Bazaar, dříve známý jako Bazaar-NG, je zástupcem DVCS systémů. Jako ostatní systémy, je i tento vyvíjen komunitou vývojářů a to pod licencí GPLv2. Jeho první verze vyšla 26. 3. 2005. Poslední dostupná verze pak vyšla 4. 8. 2013. Tento systém je od roku 2008 začleněn do GNU projektu a je psán programovacím jazykem Python a jazykem C. Ze zde již zmíněných systémů pro správu verzí je patrné, že se jedná o poměrně nový systém. Výhodou je, že Bazaar podporuje import projektů z jiných typů systémů správy verzí (např. ze Subversion). V současné době není tento systém správy verzí tolik rozšířený jako především Subversion a Git. [7]

2.2.7 Perforce

Perforce je komerční systém pro správu verzí vyvinutý společností Perforce Software. Tento systém se řadí mezi DVCS. Do centrální databáze se ukládají metadata týkající se systému (stav souborů, atributy souborů, větvení a slučování větví, změna popisů, uživatelé, skupiny). V databázi je uložen také obsah souboru jako MD5 otisk, který je použit pro ověření integrity souboru v úložišti. Samotný obsah se ukládá do adresářové struktury na serveru. Tento systém je v omezené verzi nabízen k testovacím či akademickým účelům zdarma. Nicméně na oficiálních stránkách výrobce lze dohledat, že je tento systém v komerční sféře hojně nasazovaný. [8]

2.3 Portály pro sdílený vývoj aplikací

Na základě předchozí kapitoly je zřejmé, že existuje více přístupů ke správě verzí a podle toho také existují jednotlivé nástroje, které správu verzí provádějí. A právě na základě existence různých systémů pro správu verzí byly vytvořeny portály, které podporují sdílený vývoj aplikací. Smyslem takového portálu je poskytnout prostředky pro samotný vývoj a vývojáře s cílem usnadnit samotný vývoj, správu verzí a případně i další služby. Druhým cílem je pak možnost distribuovat zde uložený projekt i běžným uživatelům, kteří si jej mohou stáhnout přes webové rozhraní.

Většina takovýchto portálů pak nabízí i další služby, které jsou určeny především pro sdílený vývoj. Mezi ně patří služba pro hlášení nových chyb a sledování těch současných (tzv. bug tracking system), jakým způsobem jsou řešeny a kdo je řeší. Vylepšenou verzí služby hlášení chyb je pak tzv. issue tracking system, který kromě správy chyb umožňuje diskuzi o dalších krocích a případných vylepšeních. Další službou, která může být pozitivním přínosem pro celý projekt, jsou manuálové stránky projektu (tzv. wiki). Zde je možné vytvářet např. stránky s nápovědou či popisem celého projektu. Samozřejmostí pak bývá určitá statistika projektu ať už v textové podobě či ve formě grafů. Z této statistiky je možno získat data nejen o počtu vývojářů, počtu stažení, počtu provedených změn, ale také třeba o změnách provedených v jednotlivé dny či o počtu přidaných a odebraných souborů. Díky webovému rozhraní je i pohodlné prohlížení zdrojových kódů a to i s možností zvýrazňování jednotlivých změn v konkrétním souboru. V případě, že je projekt sdílen na některém z těchto portálů, jsou pak nespornou výhodou právě zmíněné služby, které vývojářům usnadňují vývoj, protože pak nemusí hledat na více stránkách, ale vše mají k dispozici na jednom portálu.

2.3.1 Github

Github je webový portál pro sdílený vývoj aplikací napsaný v jazyce Ruby on Rails. Veřejně byl uveden v květnu 2008. Jako systém správy verzí používá výhradně Git. I když i zde je možné z klienta Subversion přistupovat k repositáři v systému Git. Lze použít zdarma nebo komerčně. Pro účast na vývoji je nezbytná registrace. Registrovaný uživatel si může vytvořit neomezený počet veřejných repositářů. Vytvoření privátních repositářů, u kterých lze detailně nastavit přístupová práva, je možné pouze komerčně,

resp. za poplatek. K dispozici je přehledná a propracovaná nápověda pro uživatele a programy pro správu repositáře. [9]

Přestože Github není zdaleka jediný portál pro sdílený vývoj aplikací, je mezi uživateli populární. To dokazují i statistiky hostovaných projektů, ze kterých je patrný strmý nárůst počtu repositářů, kdy v roce 2010 existoval 1 milion repositářů a na konci roku 2013 to již bylo přes 10 milionů repositářů. Mezi známé projekty hostované na tomto portálu patří jQuery, Ruby on Rails a Bootstrap.

2.3.2 Bitbucket

Bitbucket je webový portál pro sdílený vývoj aplikací, který vznikl v roce 2008. Implementován je v jazyce Python. Jako systémy správy verzí nyní nabízí Git a Mercurial. Až do roku 2011 přitom podporoval pouze Mercurial. Je zde možné si založit jak privátní, tak i veřejný repositář. Počet repositářů na uživatele není omezen, ale počet vývojářů v rámci jednoho repositáře je ve zdarma dostupné verzi limitován. Stejně jako Gihub a ostatní portály, i Bitbucket nabízí přívětivé uživatelské rozhraní pro správu repositářů, manuálových stránek, správu jednotlivých verzí, atd. Jelikož je Bitbucket určen i ke komerčním účelům, jsou přímo pro jeho použití zpracovány přehledné návody dle [10]. Mezi nejvýznamnější projekty, které jsou zde hostovány, patří Pypy, Sphinx a TortoiseHg. [11]

2.3.3 SourceForge

SourceForge je webová služba pro verzování a vývoj softwaru. Byla spuštěna v listopadu 1999 a v současnosti se řadí mezi největší služby takového typu. Aktuálně hostuje více než 260 000 projektů. Je to centrální úložiště se správou verzí pro vývojáře softwaru, kteří se podílejí na vývoji open source softwaru. Jako systém pro správu verzí lze použít CVS, SVN, Bazaar, Git a Mercurial. Právě široká podpora různých verzovacích systémů je předností tohoto portálu. K tomuto úložišti je dostupné i webové rozhraní, odkud si běžní uživatelé v případě zájmu vybraný software mohou stáhnout. Pro stažení není vyžadována registrace, ale pro účast na vývoji ano. SourceForge byl první, který nabízel takovéto služby. Mezi nejznámější projekty, které jsou zde hostovány, patří např. FileZilla, TightVNC, clamWin, TinyMCE, FreeNAS, atd. [12]

2.3.4 Google code

Webový portál Google code vznikl 17. 5. 2005 jako projekt společnosti Google pro hostování open source projektů. Nabízí více systémů správy verzí např. Subversion, Mercurial, Git, atd. a také další nástroje pro usnadnění vývoje, jak je popsáno v kapitole 2.2 (viz kapitola 2.2 Verzovací systémy). Samozřejmostí je možnost nastavení přístupových práv ke každému projektu, manuálové stránky či sekce pro stahování zdrojových kódů. Ve spojení s dalšími službami, které Google poskytuje, např. Google App Engine, se jedná o vysoce praktický systém, který umožňuje spolupracovat právě i s ostatními službami vcelku jednoduše. [13]

2.3.5 Codeplex

CodePlex je webová služba pro sdílený vývoj aplikací, samozřejmě s podporou různých systémů správy verzí. Do provozu byl uveden v květnu 2006 společností Microsoft. Mezi jeho hlavní vlastnosti patří podpora různých systémů správy verzí, např. Mercurial, Team Foundation Server, Subversion, Git. Nabízí také užitečné nástroje jako jsou diskusní fóra, wiki stránky, sledování problémů, podpora RSS, různé statistiky o vývoji atd. V porovnání se SourceForge, který v současnosti hostuje přes 260 000 projektů, CodePlex hostuje pouze přes 32 000 projektů. [14]

2.4 Projekty nad ústřednou Asterisk

Následující podkapitoly se věnují analýze existujících projektů tematicky příbuzných s ústřednou Asterisk a pochopitelně i vyvíjených komunitním způsobem. Převážně jsou analyzovány aplikace, které mají sloužit jako grafické uživatelské rozhraní pro správu ústředny Asterisk.

2.4.1 FreePBX

FreePBX je webové, grafické, uživatelské rozhraní určené pro správu telefonní ústředny Asterisk. Vývoj tohoto rozhraní započal v roce 2004 pod názvem AMP (z angl. Asterisk Management Portal). Později byl projekt přejmenován na FreePBX. Vývoj zastřešuje společnost Schmooze Com a její komunita vývojářů. Způsob vývoje tohoto projektu je komunitní s tím rozdílem, že repozitář není umístěn na portálech, které byly v této práci

dosud zmíněny, nýbrž na vlastním serveru pro tento projekt. FreePBX je volně šířeno pod licencí GNU GPLv3. [15]

Původním cílem tohoto projektu bylo vytvořit přehledné uživatelské ovládací rozhraní, které by bylo vhodné jak pro správce ústředny, tak i pro běžné uživatele, kteří by si zde mohli jednoduše spravovat své linky. Zmíněného cíle bylo dosaženo, ale tím vývoj zdaleka neskončil. Kromě běžného vývoje a oprav chyb začaly na tento projekt navazovat i další projekty a postupně se tak vytvořil jakýsi ekosystém pod značkou FreePBX. Ekosystémem je myšlena existence více projektů, které právě na FreePBX navazují a mohou jeho funkce rozšířit. Jednou z těchto navazujících myšlenek je i ta o vytvoření vlastního systému, který bude obsahovat vše potřebné pro provoz a správu telefonní ústředny. Jelikož je FreePBX navržen modulárně, myšleno je právě původní grafické uživatelské rozhraní, logickým vývojem se začaly tvořit různé moduly, které je možné do FreePBX nahrát a používat. Mezi tyto moduly patří např. pokročilý vizuální editor telefonního plánu. Mnoho z přídavných modulů je k dispozici komerčně. Pro samotnou funkčnost FreePBX ale nejsou tyto dodatečné moduly vyžadovány a toto rozhraní i ve své nativní verzi podporuje řadu více než standardních operací.

Samotné uživatelské rozhraní FreePBX je napsáno v jazycích HTML, PHP, Javascript a pochopitelně i CSS. Celá aplikace je v PHP napsána objektově. Na rozdíl od dále uvedeného Asterisk GUI může být konfigurace ukládána do MySQL databáze. Po instalaci FreePBX je poté v adresářové struktuře patrné ono rozdělení systému na samotné jádro a další přídavné moduly. Právě díky použitým technologiím, zvolenému návrhu a jeho komplexnosti, je FreePBX velice praktickým řešením, které lze snadno nasadit a provozovat v reálné praxi.

2.4.2 Asterisk GUI

Asterisk GUI je webová aplikace určená pro správu nastavení telefonní ústředny Asterisk. Její vývoj započal v roce 2006 a v roce 2013 byl oficiálně ukončen. Vývoj zastřešovala společnost Digium, která mimochodem stejně tak zastřešuje vývoj softwaru Asterisk, a její početná komunita vývojářů. Přestože byl vývoj již ukončen a oficiálně bylo doporučeno používat jiná rozhraní, zdrojové kódy jsou stále dostupné v oficiálním SVN repositáři. Asterisk GUI je šířeno pod licencí GNU GPLv2. [16]

Asterisk GUI bylo primárně určeno pro správce, který s jeho pomocí mohl nastavit prakticky jakýkoliv parametr ústředny. Tento přístup byl z hlediska funkčnosti ústředny vhodný, ale pro obyčejné uživatele mnoho pozitivního nepřinesl.

Tato aplikace je napsána v jazyce HTML a Javascript. Částečnou výhodou může být i nepotřebnost jakékoli další databáze. Jelikož stránky jsou napsány jazykem HTML a na straně serveru je nezpracovává žádný skriptovací jazyk, data si aplikace vyměňuje přímo s ústřednou Asterisk, na které musí být povoleno administrační rozhraní AMI, pomocí Javascriptu, resp. AJAXu. V posledních letech aktivního vývoje byla právě Javascriptová část přepsána do dnes moderního frameworku jQuery.

2.4.3 Asterweb

Asterweb je webová aplikace orientovaná na koncové uživatele, což je základní rozdíl oproti předcházejícím zde zmíněným aplikacím. Její vývoj započal již v roce 2005 a je publikována na serveru SourceForge. Dle dostupných informací právě na serveru SourceForge lze usuzovat, že tato aplikace není dále vyvíjena, a v průběhu let byly pouze opraveny některé chyby. Jedná se o jednoduchou aplikaci, která umí spravovat uživatele, jejich linky, hlasové schránky a příslušně upravovat telefonní plán. Z analýzy zdrojového kódu, který je psán v PHP a HTML, je zřejmý pokus o částečně objektově orientovaný přístup.

2.4.4 PAMI

PAMI (z angl. PHP Asterisk Manager Interface) je rozhraní mezi ústřednou Asterisk a PHP aplikací. Jak je patrné již z názvu, je napsáno v jazyce PHP a to objektově. Podporuje synchronní i asynchronní dotazy na AMI, což je vlastně RPC služba volání vzdálených procedur (z angl. Remote Procedure Call) na ústředně Asterisk.

2.4.5 Shrnutí existujících projektů

Z rešerše existujících projektů vyvíjených komunitou vývojářů je patrných několik závěrů. V současné době existuje jedno dále vyvíjené a široce rozšířené grafické uživatelské prostředí pro správu telefonní ústředny Asterisk. Jeho výhodou jsou použité moderní postupy, technologie a vhodně zvolený návrh uživatelského rozhraní. Jistou

nevýhodou by pak mohla být právě jeho komplexnost, protože pro malé skupiny uživatelů či malé firmy není ve většině případů potřeba zmíněná komplexnost. A právě tento pohled je základem pro praktickou část této bakalářské práce, protože ta si dává za cíl vytvořit jednoduché rozhraní, které bude disponovat pouze některými možnostmi konfigurace, bude prakticky použitelné, a bude orientováno především na malé skupiny uživatelů.

Dalším závěrem, který souvisí s předchozím, je skutečnost, že na serverech Github i Bitbucket nebyla doposud nalezena právě taková alternativa uživatelského rozhraní, která je cílem této práce. Ve většině nalezených repositářích, zabývajících se touto problematikou, byly v lepších případech nalezeny pouze moduly typu zobrazení historie hovorů či poslech hlasových zpráv, ale žádné funkční rozhraní nalezeno nebylo.

2.5 Vlastní příspěvek do existujícího projektu

Jako vlastní příspěvek do projektu vyvíjeného komunitním způsobem mohu uvést projekt, který se zabývá tvorbou interního systému správy nemovitostí pro realitní kancelář. Jelikož se jedná o komerční řešení, u kterého si zadavatel nepřeje zveřejňovat zdrojové kódy, uvádím zde pro ilustraci několik situací, které jsem osobně řešil. Pro doplnění ještě uvádím, že celý projekt je hostován na privátním serveru, na kterém běží instance Gitlabu, což je prostředí podobné Githubu, které poskytuje Git jako systém správy verzí. Kromě běžných oprav chyb, které jsou časté, spravuji přímo jednu vývojovou větev, která se zabývá výpočtem nájemného a jeho fakturací. Také jsem se svými úpravami podílel na zavádění FCP kontroléru a databázové vrstvy do této aplikace.

Konečně jako vlastní příspěvek lze považovat i praktickou část této bakalářské práce, protože je veřejně publikována na portále Github. Jak je v historii úprav na tomto portále patrné, změny v kódu probíhají, resp. probíhaly, a princip vývoje na těchto portálech nemůže změnit ani fakt, že se projektu zatím účastní jediný vývojář, resp. autor práce.

3 Použité technologie

V této kapitole jsou popsány použité technologie a postupy použité dále v praktické části této práce.

3.1 Programovací jazyky

3.1.1 PHP

PHP je multiplatformní skriptovací, programovací jazyk určený především pro programování dynamických internetových stránek. Při použití PHP jsou skripty prováděny na straně serveru a k uživateli je přenášen až výsledek jejich činnosti. V současnosti je jazyk PHP hodně rozšířený a podporuje řadu volně dostupných knihoven.

3.1.2 Javascript

Javascript je multiplatformní, objektově orientovaný, skriptovací jazyk. V současné době se zpravidla používá jako interpretovaný programovací jazyk pro webové stránky na straně klienta. Obvykle zajišťuje pokročilejší interakci mezi uživatelem a stránkou.

3.2 Ostatní technologie

3.2.1 Asterisk

Asterisk je program, který na serveru vykonává funkci telefonní ústředny. Je multiplatformní a pro základní provoz nepotřebuje žádný dodatečný hardware. Podporuje hlasové schránky, konference, fronty, práci s ID volajícího, atd.

3.2.2 MySQL

MySQL je multiplatformní databázový systém, který komunikuje pomocí jazyka SQL. Je považován za úspěšného průkopníka dvojího licencování – je k dispozici jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí. Od počátku byl optimalizován na rychlost.

3.2.3 Apache

Apache je multiplatformní webový server, jehož předností je otevřený zdrojový kód, který je distribuován pod licencí Apache License 2. Podporuje velké množství funkcí a modulů. Mnoho z nich je implementováno jako kompilované moduly rozšiřující jádro. Apache také podporuje různé programovací jazyky např. Perl, Python nebo zmíněné PHP.

3.3 Návrhové vzory

3.3.1 MVC – Model View Controller

MVC dělí aplikaci na 3 logické části tak, aby je šlo upravovat samostatně a dopad změn byl na ostatní části co nejmenší. Tyto tři části jsou Model, View a Controller. Model reprezentuje data a logiku aplikace, View zobrazuje uživatelské rozhraní a Controller má na starosti tok událostí v aplikaci a obecně aplikační logiku.

Takto lze MVC popsat ve stručnosti. Jelikož je ale tento popis do značné míry obecný, pro účely této práce bude potřebné jej upřesnit na základě reálných požadavků, možností a omezení aplikace. Zmíněné upřesnění, které je použito v praktické části této práce, je popsáno v kapitole 5.5 (viz kapitola 5.5 Návrh architektury).

3.3.2 FCP – FrontController Pattern

FCP je návrhový vzor, který v celé aplikaci vytváří jediný přístupový bod pro uživatelské požadavky. Tento přístupový bod se nazývá bootstrap a zajišťuje zpracování požadavků. Jeho běžnou součástí je nejprve nalezení konkrétní stránky, na kterou uživatel posílá svůj požadavek (tzv. routing) a poté provedení poslaného požadavku (tzv. dispatching). Výhodou tohoto návrhu je dobrá strukturovatelnost aplikace a snadná změna použitých adres.

3.3.3 Singleton

Singleton je návrhový vzor, který v celé aplikaci udržuje pouze jednu instanci dané třídy. Tato instance pak může být přístupná i globálně.

4 Nastavení ústředny

Jak je zmíněno v úvodu této práce, nejprve je třeba modifikovat současné nastavení ústředny, aby bylo možné její funkce rozšířit. Pro možnosti rozšíření je uvažována podpora IVR a dostupnost veřejných telefonních linek.

4.1 Podpora IVR

IVR, jako automatizovaný systém interaktivní hlasové odezvy, může v mnoha případech ulehčit práci operátorům a případně i volajícím. Jedná se totiž o systém, který uživateli nabízí předdefinované možnosti a uživatel mu na ně zadává odpovědi, v případě mobilního telefonu pomocí stisků jednotlivých kláves. Takovéto jednodušší systémy mohou poskytovat volajícímu ucelené informace, které potřebuje zjistit. Jako příklad jednoduššího IVR lze uvést třeba aktivaci produktů po telefonu, kdy je třeba ověřit licenční číslo, které uživatel na výzvu systému zadá. Dalším příkladem může být předpověď počasí, kdy systém uživateli nabídne sledovaná místa, a uživatel volbou zvolí požadované místo, u kterého si přeje zjistit předpověď počasí.

Mezi složitější IVR systémy lze například zařadit takové, které od volajícího získají potřebná data. Tato data zpracují a předají hovor operátorovi, který na základě právě zadaných dat může hovor dále vést s již dostupnými daty. Takovýmto zástupcem mohou být například call centra veřejných operátorů či různé marketingové agentury.

Pochopitelně existuje mnoho situací, kdy je možné systém IVR nasadit. Stejně tak jsou případy, kdy je operátor, resp. lidský kontakt, nezastupitelný. Cílem této práce je vyzkoušet jednu konkrétní implementaci v reálné případě. Jelikož se jedná o úpravu telefonního plánu, nemělo by tím být výrazně ovlivněno následné vytvoření uživatelského rozhraní, které poté bude sloužit i k úpravě zmíněného telefonního plánu.

Podpora IVR ze strany telefonní ústředny Asterisk nevyžaduje žádný přídavný modul. Vše lze provést v telefonním plánu. Zde ale opět záleží na tom, v jaké konfiguraci se právě ústředna nachází. Pokud je použita statická konfigurace, telefonní plán i příslušné kontexty jsou čteny ze souboru *extensions.conf*. V dalším případě může ústředna používat dynamickou konfiguraci. V tomto případě je pak telefonní plán uložen v databázové entitě a definice jednotlivých kontextů v souboru *extensions.conf*. Logický

rozdíl mezi jednotlivými plány není. Pouze pro každý způsob konfigurace jsou uloženy na různých místech, ale syntaxe a logická stránka příkazů zůstávají zachovány.

Pro účel demonstrace funkčního IVR byla vybrána aplikace, která uživateli nabídne seznam míst a uživatel si volbou zvolí, pro jaké místo si přeje slyšet předpověď počasí. Tento případ vyžaduje vyhradit jednu telefonní linku v interním kontextu, pod kterou bude dostupná tato služba (zvolena linka ***17**), a dále vytvořit nový kontext, ve kterém bude řešeno samotné IVR. Nyní následuje ukázka kódu se zápisem řešeného IVR (viz ukázka kódu 1).

```
[internal]
exten => _*17,1,Goto(weather,s,1)

[weather]
exten => s,1,Background(for&weather&in-the
                    &atlanta&press-1)
exten => s,2,Background(or&for&weather
                    &in-the&boston&press-2)
exten => s,3,WaitExten(5)
exten => s,4,Goto(1)
exten => i,1,Playback(sorry)
exten => i,2,Hangup()
exten => _1,1,Playback(wind&and)
exten => _1,2,SayNumber(21)
exten => _1,3,Playback(celsius)
exten => _1,4,Hangup()
exten => _2,1,Playback(rain&and)
exten => _2,2,SayNumber(19)
exten => _2,3,Playback(celsius)
exten => _2,4,Hangup()
```

Ukázka kódu 1: IVR předpověď počasí

Na tomto příkladu je použita statická konfigurace ústředny. Jelikož ale ústředna pro další potřeby této práce pracuje v dynamické konfiguraci, je zde tato ukázka spíše kvůli logickému pohledu na toto IVR, který zůstává stejný i v dynamické konfiguraci. Pouze je příslušně transformován do databázové entity spravující telefonní plán. Tato transformace, resp. SQL kód, je v celé podobě uložen na přiloženém CD se zdrojovými kódy.

4.2 Veřejné linky

Jak je uvedeno dále v kapitole zabývající se požadavky vytvářeného rozhraní (viz kapitola 5.1 Definice požadavků), bude nutné změnit konfiguraci ústředny tak, aby umožňovala přenášení hovorů do ostatních sítí a dokázala také hovory z těchto sítí přijímat. Tímto krokem je myšleno nastavení ústředny takovým způsobem, aby komunikovala s další ústřednou, která zmíněné propojení sítí poskytuje. Tento krok byl již v ročníkové práci řešen, ale provedené řešení má značné omezení. Toto omezení spočívá v možnosti uskutečňovat současně pouze jeden hovor s jinou sítí. Jinými slovy, naše ústředna má přidělené pouze jedno veřejně dostupné telefonní číslo, resp. linku s veřejným číslem, na kterém lze provádět současně pouze jeden hovor. Analogicky, pokud by byla potřeba v interní síti přidělit veřejné číslo konkrétnímu telefonu, na ostatní by již žádné další číslo nezbylo.

Popsaný nedostatek předešlého řešení lze odstranit tak, že bude na zprostředkovatelské ústředně rezervováno více veřejně dostupných čísel se samostatnými linkami. Tím bude zajištěna flexibilita v případě potřeby změny počtu veřejných čísel. Tato změna si ale vyžádá úpravu konfigurace ústředny. Cílem je nakonfigurovat ústřednu takovým způsobem, aby se na zprostředkovatelské ústředně pouze rezervovala potřebná čísla a na naší ústředně nebyl vyžadován jediný zásah do konfigurace. Právě kvůli tomuto požadavku je nutné upravit stávající konfiguraci v souboru *sip.conf*. Aktuálně je zde totiž potřeba každou veřejnou linku registrovat odděleně. Toto nastavení lze provést dle ukázky kódu 2, která zajistí spojení se zprostředkující ústřednou a příchozí hovory bude směrovat do kontextu **incoming**, kde budou obslouženy.

```
[from-odorik]
type=peer
host=sip.odorik.cz
context=incoming
```

Ukázka kódu 2: registrace ústředny

V tomto okamžiku je ústředna připravena přijímat hovory, které jí pošle zprostředkující ústředna z veřejně dostupných čísel, které jsou pro naši ústřednu registrovány. Stejně tak je možné hovory vytvářet a posílat přes zprostředkující ústřednu ke koncovým zařízením veřejné sítě. Odchozí hovory jsou zpracovávány v kontextu **outgoing**.

Pro potřeby této práce byly rezervovány dvě veřejné linky. Konkrétně se jedná o linku **488 588 286** a **499 599 759**. Smyslem je demonstrovat funkčnost vyvíjeného řešení na více linkách a z tohoto pohledu je třeba použít alespoň dvě linky.

4.3 Přesměrování na mobilní telefon

V průběhu zpracování této práce byla ještě zvažována myšlenka rozšířit funkce ústředny o službu přesměrování na mobilní telefon. Toto rozšíření mělo umožňovat při nedostupnosti volané lokální linky přesměrovat hovor na předem zadaný mobilní telefon účastníka. Celá tato myšlenka by vyžadovala dvě zásadní úpravy. Nejprve by bylo nutné zásadně změnit strukturu interního kontextu v telefonním plánu. V současné konfiguraci (viz ukázka kódu 3) je obsah interního kontextu zpracováván především regulárními výrazy, které určují konkrétní linky, na které se volá. Při směrování nedostupných hovorů na mobilní telefon by bylo zapotřebí v interním kontextu explicitně definovat každou linku a poté její přesměrování. Z toho je patrné, že by se interní kontext rychle plnil a bylo by obtížnější jej udržovat.

```
[internal]
exten => _ZXXX,1,Dial(SIP/${EXTEN},30,m)
```

Ukázka kódu 3: telefonní plán

Druhým bodem, nad kterým by bylo třeba konkrétně uvažovat, je volba politiky přidělování veřejných čísel. V tomto případě by bylo praktické, aby měla každá interní linka k dispozici i veřejné číslo. Tento požadavek vyplývá z možné situace, kdy bude interní linka volat jinou interní linku, která nebude odpovídat. V případě, že dojde k přesměrování na mobilní telefon, a volající linka by neměla přidělené veřejné číslo, hovor by nebyl spojen.

Na základě uvedených skutečností bylo rozhodnuto, že toto rozšíření nebude realizováno.

5 AsteriskAdmin

Tato kapitola popisuje jednotlivé vývojové kroky při vytváření ovládacího rozhraní pro telefonní ústřednu Asterisk. Mimo základních kroků, nutných pro samotný vývoj, jsou zde také popsány konkrétní situace, které bylo nutné vyřešit, a jejich řešení.

5.1 Definice požadavků

Cílem je vyvinout webovou aplikaci pro správu telefonní ústředny, která bude poskytovat dále popsané funkce. Prvním požadavkem je základní ovladatelnost napříč různými operačními systémy a webovými prohlížeči. Pod pojmem základní ovladatelnost se rozumí zachování a proveditelnost všech dále specifikovaných funkcí. Napříč různými prohlížeči jsou povolené drobné odlišnosti ve zobrazení webové stránky aplikace, avšak funkčnost musí být zachována a být identická. Dále tato aplikace nesmí vyžadovat instalaci programů třetích stran, resp. pluginů do prohlížeče. Vzhledem k současné konfiguraci ústředny budou veškeré změny v její konfiguraci prováděny v příslušných databázových entitách. Pro svoje potřeby může aplikace použít adresář, ve kterém bude nahrána, a dále libovolný počet databází na databázovém serveru, na kterém je provozováno MySQL. Aplikace by dále měla využívat uživatelsky čitelné URL.

Do aplikace mají mít přístup dva typy uživatelů, a to správce a běžný uživatel. Oba dva typy uživatelů mají mít dostupné funkce, které se týkají samotného přístupu do aplikace. Jmenovitě to jsou: přihlášení, odhlášení, změna zapomenutého hesla, možnost úpravy vlastního uživatelského účtu a prohlížení historie provedených událostí uživatelského účtu. Běžný uživatel by dále měl mít k dispozici možnost vytvářet, upravovat a mazat vlastní telefonní linky. K těmto linkám navíc přiřazovat hlasové schránky s možností prohlížet a mazat jejich obsah. Vzhledem k možnostem ústředny je také požadována funkce propojení vlastní telefonní linky s veřejně dostupným telefonním číslem. Jelikož jsou zaznamenávány veškeré provedené hovory, aplikace by měla umožnit zobrazit seznam provedených hovorů podle jednotlivých linek přihlášeného uživatele. Posledním požadavkem pro uživatele je zobrazení seznamu existujících telefonních linek v rámci ústředny a jejich vlastníků. Toto zobrazení nemá být jakkoli omezeno.

Dosud zmíněné funkce by měly být dostupné pro běžného uživatele i správce. Správce by měl mít k dispozici i následující funkce, které souvisejí se správou této aplikace i ústředny. Mezi tyto funkce patří možnost vytvořit nový uživatelský účet, zobrazení seznamu existujících uživatelů a možnost vytvářet či mazat veřejně dostupná telefonní čísla. Další funkcí je správa telefonního plánu ústředny. V tomto plánu by měly být dostupné operace vytvoření, úpravy a smazání jednotlivých položek.

5.2 Analýza požadavků

Na základě požadavků definovaných v předchozí kapitole je zřejmých několik skutečností. První skutečností je relativní neomezenost v otázce operačního systému a webového serveru, na kterém bude aplikace provozována. Jediným drobným omezením v tomto směru je požadavek na uživatelsky čitelné URL. Vzhledem k tomuto požadavku a také k osobní zkušenosti byl pro provoz této aplikace vybrán operační systém CentOS a webový server Apache. Výhodou webového serveru Apache je dostupnost modulu, který se zabývá uživatelsky čitelnými URL. Na straně serveru také nebyl přesně definován skriptovací jazyk, který je podporován. Jelikož byl ale výběr operačního systému a webového serveru volný, v logické vazbě na tuto kombinaci byl vybrán skriptovací jazyk PHP.

Další skutečností je požadavek na přenositelnost aplikace mezi různými prohlížeči. To znamená, že na straně klienta musí být použity standardizované technologie pro webové stránky a musí se v řešení uvažovat o jednotlivých odlišnostech současných prohlížečů. Webové stránky budou tudíž napsány v jazyce HTML. Jako doplnění je možné použít i jazyk Javascript. Potíží při použití jakýchkoli skriptů v jazyce Javascript může být jeho deaktivace na straně klienta. Tento problém je možné řešit více způsoby, avšak pro tuto práci bylo rozhodnuto naprogramovat veškerou funkčnost pomocí HTML a dalších potřebných jazyků, a jazyk Javascript použít pouze na doplnění stránky o diskrétní grafické efekty. Z toho je zřejmé, že případná deaktivace Javascriptu nebude mít na funkčnost aplikace vliv.

Na základě požadovaných funkcí aplikace byl sestaven diagram případů užití (viz Příloha A), který popisuje základní strukturu aplikace. Z tohoto diagramu je patrná existence dvou typů uživatelů. Obecným typem je běžný uživatel, který má k dispozici

správu vlastních telefonních linek, která zahrnuje i přidělení a uvolnění veřejného telefonního čísla, nastavení hlasové schránky a přehled registrovaných zařízení k upravované lince. V rámci procházení hlasových schránek může uživatel jednotlivé zprávy mazat. Mezi další funkce patří přehled hovorů k jednotlivým linkám, telefonní seznam existujících linek a historie událostí k přihlášenému účtu. Poslední kategorií dostupných funkcí je řízení přístupu, do kterého spadá přihlášení, odhlášení, úprava uživatelského profilu, vytvoření nového účtu, nastavení zapomenutého hesla a také vygenerování žádosti o obnovu zapomenutého hesla. Tato žádost je odesílána prostřednictvím emailu na adresu uvedenou při registraci. Druhým typem uživatele je správce, který pouze rozšiřuje možnosti běžného uživatele. Mezi rozšiřující funkce patří vytváření nových uživatelů, správa veřejných linek a také správa telefonního plánu. Při vytváření nových uživatelů je na zadanou adresu odeslán email, kterým registrující se uživatel dokončí vytvoření svého profilu.

5.3 Návrh databáze

Na základě definovaných požadavků je možné použít pro interní potřeby vyvíjené aplikace databázi. Mezi tyto potřeby bezpochyby patří uchovávání přístupových údajů uživatelů. Dále by zde také měla být uložena historie provedených událostí jednotlivých uživatelů a také historie interních chyb samotné aplikace.

Pro vhodné navržení struktury databáze je třeba zohlednit skutečnost, že nastavení ústředny jsou taktéž umístěna v databázi. Z toho vyplývají dva možné způsoby řešení. Prvním z nich je sloučení obou databází do jedné. Toto řešení by nikterak neovlivnilo současnou potřebu ústředny pracovat s databází, jelikož by všechny její entity zůstaly zachovány ve stávajícím stavu a byly by přidány další entity pro potřeby aplikace. V aplikaci samotné by tento přístup usnadnil práci s databází, jelikož by bylo potřebné pouze jedno spojení s jednou databází. Druhým uvažovaným přístupem je oddělení obou databází. To znamená zachování stávající databáze s konfigurací ústředny a vytvoření nové databáze pro vlastní aplikaci. Z popisu tohoto přístupu je patrná potřeba aplikace vlastnit dvě připojení na databázi a správně rozlišovat, kdy se jaké použije. Jelikož při analýze požadavků nebyla zjištěna jakákoli skutečnost, která by znemožňovala navrhnout databázi prvním způsobem, bylo zvoleno sloučení databází.

Na základě požadavků byl vytvořen návrh databázových entit (viz Příloha B), které rozšíří stávající databázi. Novými entitami v databázi jsou:

- uživatel (user) – ukládání přístupových údajů uživatelů,
- uživatelská historie (system_log_activity) – historie provedených operací,
- systémová historie (system_log_internal) – zaznamenání interních chyb,
- registr nových uživatelů (ucr) – ukládání požadavků registrace.

Přidání těchto entit do databáze tedy neovlivní stávající potřebu ústředny a navíc bude zjednodušen přístup ke konfiguraci samotné, jelikož vyvíjená aplikace bude požadovat pro všechny úkony pouze jedno připojení k jedné databázi.

5.4 Formát URL

Pod termínem uživatelsky čitelné URL, který je zmíněn v kapitole 5.1 (viz kapitola 5.1 Definice požadavků), se rozumí taková URL adresa, která je dobře čitelná a zapamatovatelná pro běžného uživatele. Pro přesné umístění zdrojů v internetu se požívá URL, jejíž součástí je protokol, doména druhého řádu, generická doména, port, umístění v rámci serveru a formulářová data. Právě až poslední položka, tedy formulářová data, jsou tou částí URL, která lze specificky upravit podle potřeb tak, aby byla zapamatovatelnější. Na následujících URL adresách jsou názorně zobrazeny rozdíly mezi standardním tvarem a uživatelsky čitelným tvarem:

- http://server.cz/**index.php?modul=uzivatel&akce=prihlaseni**
- http://server.cz/**uzivatel/prihlaseni/**.

Pro možnost používání takto upravených adres je na webovém serveru Apache nutný nainstalovaný modul **mod_rewrite**. Tento modul dovolí modifikovat přenášené URL adresy podle tvarů, které mu budou explicitně nastaveny. Toto nastavení je možné provést v souboru *htaccess* v kořenovém adresáři aplikace, pokud je tato možnost na serveru povolena. Následující kód (viz ukázka kódu 4) demonstruje situaci, kdy uživatel zadá do prohlížeče pro něj čitelnou adresu (**/uzivatel/prihlaseni/**) a ta je na webové serveru přeložena na tvar s otazníkem a parametry s rovnítkem. Po zpracování a odeslání odpovědi uživatel uvidí ve svém prohlížeči opět pro něj čitelnou adresu.

```
RewriteCond %{REQUEST_URI} ^/([a-z]+)/([a-z]+)/$
RewriteRule ^([a-z]+)/([a-z]+)/$ /index.php&m=$1&a=$2
```

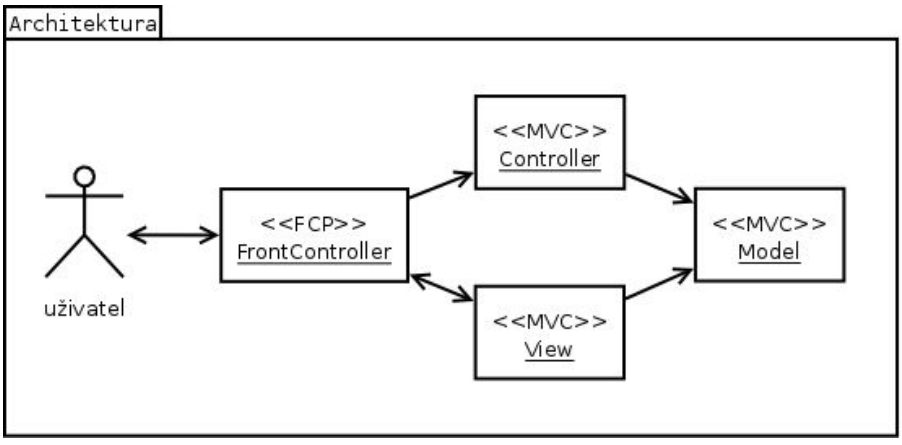
Ukázka kódu 4: pravidlo pro přepsání URL

Celé to tedy funguje jako jakési maskování, kdy uživatel zadává adresu, která je poté transformována a vyhodnocena standardně pomocí polí GET či POST. O správné převedení do těchto polí se stará webový server podle pravidel definovaných v souboru *htaccess*.

5.5 Návrh architektury

Obecným předpokladem při zadávání většiny aplikací bývá nutnost modifikace kódu v průběhu času. Aby byl program dobře modifikovatelný, musí být dobře navržena jeho základní struktura, nebo-li architektura. A to je jedním z cílů této bakalářské práce, vhodně navrhnout a na základě návrhu vytvořit aplikaci. Z uvedeného důvodu bude aplikace vytvořena objektově orientovaným způsobem s využitím několika návrhových vzorů.

Základním stavebním prvkem jsou návrhové vzory MVC a FCP. Jejich stručné definice jsou uvedeny v předchozí kapitole, která popisuje použité technologie (viz kapitola 3.3 Návrhové vzory). Aby bylo možné tyto návrhové vzory použít a provést jejich konkrétní implementaci, nejprve je nutné zamyšlení nad jejich vzájemným propojením. Toto propojení je zobrazeno na následující ilustraci (viz obr. 1).



Ilustrace 1: Architektura aplikace

Veškeré požadavky jsou směrovány přes FrontController, který zajistí jejich zpracování v příslušných objektech. Aby toto zpracování mohl iniciovat, musí mít odkaz na instance konkrétních objektů, které danou operaci mají provést. Těmito objekty, které provádějí zpracování, jsou objekty typu Controller a View. Oba tyto typy objektů mají dále odkaz na stejnou instanci datového modelu typu Model. Z ilustrace je patrné, že mezi objekty Controller a View neexistuje přímá vazba. To z tohoto důvodu, aby se vzájemně neovlivňovaly.

V tuto chvíli je jasné, jak se mezi sebou dané objekty volají. Ale ještě je třeba zabývat se otázkou, jak FrontController zjistí, kterým z objektů má předat zpracování události, a jak má získat odkazy na instance daných objektů. Ještě před samotným vytvořením daných objektů si FrontController vytvoří objekt typu Router, který je určen k dynamickému směrování požadavků. Router podle parametrů požadavku vybere správné objekty k vytvoření a předá FrontControlleru zpět informace o tom, které objekty si má vytvořit. FrontController dané objekty vytvoří a iniciuje zpracování požadavku. Pro objasnění druhé otázky je určen sekvenční diagram aplikace (viz Příloha C), který zobrazuje životnost webové stránky a v rámci této životnosti pak vytváření a volání jednotlivých objektů. Z diagramu i dosavadního popisu je zřejmé, že po vytvoření objektů typu Model, View a Controller ve FrontControlleru, FrontController nejprve zavolá vytvořený Controller. Ten má za úkol ověřit, zdali je akce proveditelná a uživatel splňuje všechny požadované vlastnosti. Pokud jsou tyto podmínky splněny, může se provést požadavek na změnu dat v datovém modelu. Po provedení operací v Controlleru se běh aplikace vrací zpět do FrontControlleru, který iniciuje objekt typu View. Ten má za úkol vygenerovat a vrátit obsah HTML stránky na základě datového modelu a parametrů v požadavku od uživatele. Po vykonání View je zpracování stránky dokončeno.

Z důvodů zmíněné modularity jsou objekty typu Model, View a Controller definovány jako abstraktní třídy. Pro každou skupinu operací, které vzájemně nějakým způsobem souvisí, jsou poté vytvářeni potomci těchto tříd, kteří zavádějí vlastní implementaci požadovaných funkcí.

Pro každou proveditelnou operaci existuje v konkrétních potomcích tříd Controller a View metoda, která operaci provádí. Z hlediska průběhu zpracování stránky

(viz Příloha C) je vhodné umístit případné omezující podmínky pro provedení této operace do metody v Controlleru. Ten se totiž provádí první a pokud se při jeho provádění zjistí, že je požadavek z nějakého důvodu neplatný, může se zablokovat provádění View. Na stejném místě lze také rozlišovat, jakou metodou byly parametry stránce předány a tudíž jak mají být zpracovány (jestli se jedná o formulář či zobrazení).

V tomto okamžiku je možné použít existující framework a aplikaci ve vybraném vytvořit. Výhodou tohoto způsobu je připravenost většiny dostupných frameworků a relativní jednoduchost tvorby. V tomto případě se vývojář nemusí příliš zabývat již vytvořenými komponenty a věnuje se především logické části aplikace. Z právě uvedeného důvodu tento postup pro tvorbu aplikace AsteriskAdmin nebyl zvolen.

5.6 Správa uživatelů

Mezi základní funkce správy uživatelů patří vytváření nových uživatelů a správa jejich účtů. Jelikož je aplikace navržena takovým způsobem, který umožňuje provedení jakékoli operace pouze přihlášeným uživatelům, další funkcí tak musí být zajištění přístupu do aplikace.

Přístupové údaje každého uživatele jsou uloženy v databázové entitě uživatel (user). Mezi základní zde uložené údaje patří:

- email – jednoznačný identifikátor uživatele,
- heslo – otisk pomocí funkce SHA256 ve formátu **email:heslo**,
- typ uživatelské účtu – správce nebo běžný uživatel.

Heslo je ukládáno ve formě otisku i z důvodu, aby i pro správce databáze, který si jednotlivá data může zobrazit, bylo toto heslo nerozpoznatelné. V případě útoku na databázový server, kdy by se útočníkovi podařilo získat data z databáze, by byl ve stejné situaci jako správce, protože uložené heslo by pro něj nebylo jednoduché v dostupné době rozluštit.

Registraci nového uživatele může provést pouze správce ústředny. V takovém případě potřebuje znát pouze email, pod kterým se bude nový uživatel přihlašovat. Po odeslání registračního formuláře se do databázové entity registru nových uživatelů (ucr) vloží záznam o provedené registraci konkrétní emailové adresy. Na tuto adresu je následně

odeslán email s aktivačním kódem a instrukcemi. Registrující se uživatel při vstupu na stránku vytvoření nového uživatele pomocí aktivačního kódu prokáže svoji způsobilost k registraci a vyplní si potřebné údaje pro vytvoření účtu. Hlavním takovýmto údajem je heslo. Mezi další volitelné údaje patří jméno a příjmení. Výhodou tohoto přístupu je možnost nastavovat osobní údaje pouze pod svým uživatelským účtem. To znamená, že správce takovéto zásahy do údajů běžných uživatelů provádět nemůže.

Změna hesla je prováděna obdobným způsobem jako registrace nového uživatele. Jediný rozdíl spočívá v tom, že zapomenuté heslo si může uživatel obnovit sám, zadáním emailové adresy do formuláře pro obnovu hesla. Následně je mu zaslán email s instrukcemi a potvrzujícím kódem, kterým se musí prokázat, aby mu bylo umožněno heslo nastavit na nové.

5.7 Veřejné linky

Veřejné linky, které jsou registrovány na zprostředkující ústředně pro naše použití, je nutné zaregistrovat v telefonním plánu této ústředny. Tuto registraci provádí správce ústředny prostřednictvím příslušného formuláře zadáním veřejně dostupného telefonního čísla do kontextu **incoming** v telefonním plánu. Tím se definují veřejně dostupná telefonní čísla, která si pak běžní uživatelé mohou přiřazovat ke svým interním linkám. V okamžiku přiřazení k interní lince se provedou dvě operace. První z nich je uložení informace o novém vlastníkovi veřejné linky do kontextu **incoming**, které je znázorněno na následující ukázce kódu 5.

```
[incoming]
exten => _00420499599759,1,Dial(SIP/1005)
```

Ukázka kódu 5: definice veřejné linky

V tomto okamžiku je možné na interní lince přijímat hovory z veřejně dostupného čísla. Aby bylo možné hovory z interního čísla vytvářet, a ty se poté zobrazovaly volajícímu pod správným veřejným číslem, je potřeba korektně nastavit číslo volajícího. Toto nastavení je druhým krokem při přidělení veřejného čísla k interní lince a je prováděno v kontextu **outgoing**. Jeho základní strukturu demonstruje ukázka kódu 6. Pro zjednodušení je uváděno **volane_cislo**, které je potřeba nahradit řetězcem **_00420[23456789]XXXXXXXX** (postihuje všechny sítě v ČR), a také **volajici**, které je

nutné nahradit řetězcem **CALLERID(num)**.

```
[outgoing]
exten => volane_cislo,1,Goto(4)
exten => volane_cislo,2,Dial(SIP/${EXTEN}@to-odorik)
exten => volane_cislo,3,Hangup()
exten => volane_cislo,4,Gotoif(volajici='1005'?5:7)
exten => volane_cislo,5,Set(volajici='499599759')
exten => volane_cislo,6,Goto(2)
```

Ukázka kódu 6: nastavení veřejného čísla

5.8 Ošetření zranitelností

U webových aplikací existují tři základní druhy zranitelností, které je vhodné zahrnout již do návrhu samotné struktury vytvářené aplikace. Mezi tyto typy útoků patří XSS (viz [17]), SQL Injection (viz [18]) a CSRF (viz [19]). K ošetření těchto zranitelností existují postupy, které minimalizují možnost provedení příslušných útoků.

Zranitelnost XSS a SQL Injection jsou si do jisté míry podobné. Ke svému účelu potřebují modifikovaný vstup od uživatele. Jejich rozdílem pak je místo kam se škodlivý kód předává a co má způsobit. Zranitelnost typu XSS cílí na Javascriptový kód stránky, kdy v případě nevhodně ošetřených vstupů může být zadaný kód proveden. A pokud k takovému provedení dojde, existuje riziko zneužití vlastních dat aplikace. Naopak zranitelnost typu SQL Injection využívá upravený vstup uživatele k získání dat z databáze, kdy lze pomocí špatně ošetřených vstupů sestavovat libovolné dotazy, které budou následně nad databází provedeny. Mezi základní techniky ošetření uživatelských vstupů se používají regulární výrazy, odstranění bílých znaků po stranách řetězce a escapovaní znaků. Všechny zmíněné techniky jsou v aplikaci použity.

Další zranitelností je CSRF, což je útok spočívající v tom, že uživatel je donucen navštívit stránku, která skrytě vykoná útok na webovou aplikaci, kde je uživatel zrovna přihlášen. Ošetřením této zranitelnosti je generování jedinečného kódu pro každý formulář, který uživatel může odeslat a po odeslání kontrolovat poslaný a předem vygenerovaný kód. Tyto kódy by se měly shodovat. V opačném případě je možné, že se

jedná o útok na webovou stránku. Stejně jako předchozí zranitelnosti, je i tato zranitelnost v aplikaci náležitě ošetřena.

5.9 Jazykové mutace

V definici požadavků na tuto aplikaci nebylo toto rozšíření explicitně definováno. Přes to z důvodu možné budoucí rozšiřitelnosti bylo vhodné začlenit tento modul již do samého základu aplikace. Principiálně se jedná o možnost výběru jazyka, ve kterém bude aplikace zobrazena uživateli. Každý uživatel tak má možnost volby jazyka webové stránky, který aplikace nabízí. Tato volba je uložena spolu s dalšími uživatelskými údaji v databázové entitě uživatel (user) a tudíž je dostupná po opětovném přihlášení.

Dostupné jazyky jsou uloženy v adresáři *./app/locale*. Každý jazyk je reprezentován vlastní třídou, která je potomkem abstraktní třídy **Translator**. V této rodičovské třídě jsou umístěny klíče, které definují jednotlivé překládané řetězce. Úkolem potomků této abstraktní třídy je přiřadit ke každému klíči řetězec přeložený do konkrétního jazyka a také přepsat metodu, která vrací přeložený řetězec k zadanému klíči.

Samotný překlad všech řetězců je poté prováděn pomocí statického objektu typu **Translate**, který je navržen podle návrhového vzoru Singleton, a přes který je možné získávat přeložené řetězce pomocí jejich klíčů.

5.10 Testování

Testování obecně by mělo patřit k běžným součástem vývoje. Pro každý projekt nemusejí být vhodné všechny testovací metodiky. Vždy záleží na přesném zadání projektu, konkrétní implementaci a možnostech vývojového týmu. Je patrné, že dobře navržený a otestovaný kód má větší potenciál na bezchybný provoz, než kód, který žádným testováním neprošel.

V rámci vytvářené aplikace byly pro účely testování vybrány dva druhy testů, konkrétně testování jednotek (viz [20]) a funkční testy (viz [21]). Jelikož je testování jednotek převážně záležitost lokálního charakteru ve smyslu testování funkcí jednotlivých objektů, po otestování byla každá funkce opatřena příslušným komentářem. Tento komentář běžně obsahuje informace o vstupech, výstupech a jiných skutečnostech, které mohou při použití této funkce nastat. Funkční testy byly prováděny průběžně

prostřednictvím webových stránek aplikace s cílem ověřit její funkčnost z pohledu uživatele. Veškeré chyby, které byly tímto způsobem odhaleny byly následně také opraveny.

5.11 Licence

Pro následné uvolnění aplikace AsteriskAdmin na portál Github bylo nutností zvolit licenci, pod kterou bude dále šířena. Jelikož od počátku vývoje bylo uvažováno o vytvoření svobodně dostupné aplikace s minimálními omezeními, byla zvolena licence BSD. Ta vyžaduje pouze uvedení autora a informace o licenci, spolu s upozorněním na zřeknutí se odpovědnosti za dílo.

5.12 Dostupnost

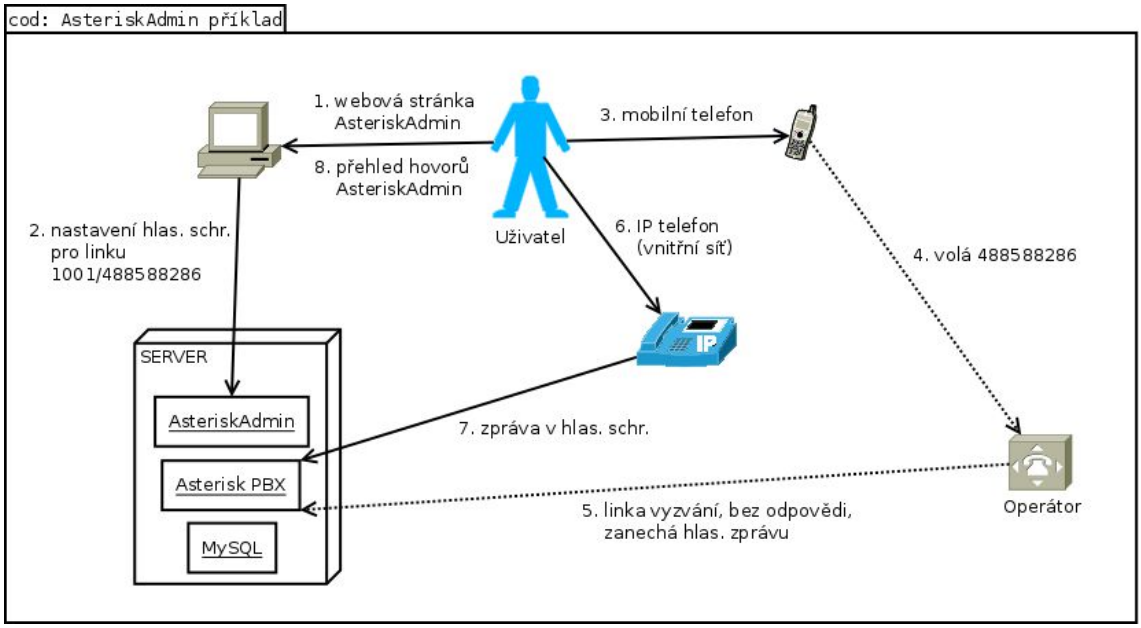
Vyvinutá aplikace byla zveřejněna na portále Github a je dostupná na adrese **https://github.com/mpi77/AsteriskAdmin**. Jelikož Github je portál pro sdílený vývoj aplikací s mezinárodní dostupností, veškeré zde uváděné kódy a komentáře jsou psány v anglickém jazyce. V rámci manuálových (wiki) stránek tohoto projektu bylo vytvořeno několik stránek, které stručně popisují problematiku projektu. Mezi těmito stránkami je například instalace telefonní ústředny nebo způsob, jak je řešen překlad do různých jazykových verzí této aplikace.

Na přiloženém CD jsou také k dispozici ukázky veškerých stránek, které tato aplikace obsahuje. Tyto ukázky jsou uloženy ve složce *./ukazky*. Mezi zařazené ukázky patří i ukázka titulní strany aplikace (viz Příloha D).

5.13 Provázanost systémů

Závěrem této kapitoly lze poznamenat, že vytvořená aplikace doplňuje telefonní ústřednu a do jisté míry přidává i možnost pro běžného uživatele se s provozem ústředny seznámit a využívat tak pohodlně jejích služeb. Provázanost telefonní ústředny a vytvořené aplikace AsteriskAdmin je zobrazena na následující ilustraci (viz obr. 2). Uživatel si přes aplikaci AsteriskAdmin nastaví hlasovou schránku pro svoji interní linku, která má přidělené i veřejně dostupné telefonní číslo. V dalším kroku uživatel prostřednictvím mobilního telefonu zavolá na uvedené veřejné číslo. Jelikož jej nikdo

nezvedne, zanechá zde hlasovou zprávu. Následně si uživatel pomocí IP telefonu ve vnitřní síti zavolá na ústřednu a vyzvedne si zde uloženou hlasovou zprávu. Navíc se může ještě podívat přes aplikaci AsteriskAdmin do zmiňované hlasové schránky a také na přehled provedených hovorů.



Ilustrace 2: Provázanost systémů

Závěr

V rámci práce jsem se seznámil s prostředky pro sdílený vývoj aplikací. S pomocí těchto prostředků jsem poté vytvořil webovou aplikaci pro správu nastavení telefonní ústředny Asterisk. V průběhu práce jsem provedl dílčí změny ve stávající konfiguraci ústředny, které měly poskytnout jejím uživatelům další funkce.

Výsledkem provedené rešerše existujících projektů na portálech pro sdílený vývoj aplikací bylo nenalezení žádného tematicky příbuzného projektu menšího rozsahu, který by poskytoval omezenou množinu potřebných funkcí pro konfiguraci ústředny. Právě tato skutečnost byla jedním z důvodů vytvoření vlastní aplikace AsteriskAdmin.

Před samotnou tvorbou aplikace proběhlo seznámení s dostupnými nástroji pro správu verzí a také jednotlivými portály, které nabízejí možnost komunitního vývoje. Na portále Github byl poté založen projekt, který slouží k publikaci jednotlivých verzí vyvíjené aplikace.

Aplikace AsteriskAdmin určená pro správu telefonní ústředny, tak jak byla vytvořena, je funkční a dále použitelná. Mezi její přínosy patří snadná ovladatelnost pro koncové uživatele a díky vhodnému návrhu i její modularita. Po provedení návrhu aplikace AsteriskAdmin bylo možné použití existujícího frameworku nebo vytvoření vlastního řešení, které se bude muset zabývat známými problémy již od nejnižších stupňů vývoje a tyto problémy řešit. Pro tvorbu byl zvolen druhý způsob, ze kterého vyplývá vyšší složitost a potřeba orientace v dané problematice.

Díky způsobu návrhu a použití vhodných technologií, mezi které se řadí mj. použité návrhové vzory, je aplikace AsteriskAdmin nejen že použitelná, ale i dále rozšiřitelná a pro ostatní vývojáře je i lehce čitelný a dokumentovaný kód hlavním přínosem.

Seznam použité literatury

[1] BUREŠ, Martin. Programování služeb a infrastruktury pro přenos hlasu po IP sítích. Liberec, 2013. Ročníkový projekt. Technická univerzita v Liberci.

[2] CHACON, Scott. Pro Git. New York: Apress, c2009, xxi, 265 s. ISBN 978-1-4302-1833-3.

[3] GNU RCS 5.9.2. GNU RCS Manual - GNU Project - Free Software Foundation (FSF) [online]. 2013 [cit. 2014-04-25].

Dostupné z: http://www.gnu.org/software/rcs/manual/rcs.html

[4] CVS. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2014 [cit. 2014-04-25].

Dostupné z: http://cs.wikipedia.org/wiki/CVS

[5] Apache Subversion. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2013 [cit. 2014-04-25].

Dostupné z: http://cs.wikipedia.org/wiki/Apache_Subversion

[6] Mercurial. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2014 [cit. 2014-04-25].

Dostupné z: http://cs.wikipedia.org/wiki/Mercurial

[7] GNU Bazaar. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2014 [cit. 2014-04-25].

Dostupné z: http://en.wikipedia.org/wiki/GNU_Bazaar

[8] Perforce. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2013 [cit. 2014-04-25].

Dostupné z: http://cs.wikipedia.org/wiki/Perforce

[9] GitHub. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2014 [cit. 2014-04-25].

Dostupné z: http://en.wikipedia.org/wiki/GitHub

[10] Bitbucket Documentation [online]. 2014 [cit. 2014-04-25].

Dostupné z: https://confluence.atlassian.com/display/BITBUCKET/

[11] Bitbucket. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2014, 20.1.2014 [cit. 2014-04-25].

Dostupné z: http://en.wikipedia.org/wiki/Bitbucket

[12] SourceForge. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2014 [cit. 2014-04-25].

Dostupné z: http://en.wikipedia.org/wiki/SourceForge

[13] Google code. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2014 [cit. 2014-04-25].

Dostupné z: http://en.wikipedia.org/wiki/Google_Code

[14] CodePlex Information and Discussion. CodePlex - Open Source Project Hosting [online]. 2012 [cit. 2014-04-25].

Dostupné z: https://codeplex.codeplex.com/wikipage?title=About%20the%20CodePlex%20Site&referringTitle=Documentation

[15] FreePBX. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2014 [cit. 2014-04-25].

Dostupné z: http://en.wikipedia.org/wiki/FreePBX

[16] Asterisk GUI. Asterisk Project Wiki [online]. 2013 [cit. 2014-04-25].

Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Asterisk+GUI

[17] Cross-site Scripting. In: OWASP [online]. 2014 [cit. 2014-04-25].

Dostupné z: https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29

[18] SQL Injection. In: OWASP [online]. 2014 [cit. 2014-04-25].

Dostupné z: https://www.owasp.org/index.php/SQL_Injection

[19] Cross-Site Request Forgery. In: OWASP [online]. 2013 [cit. 2014-04-25].

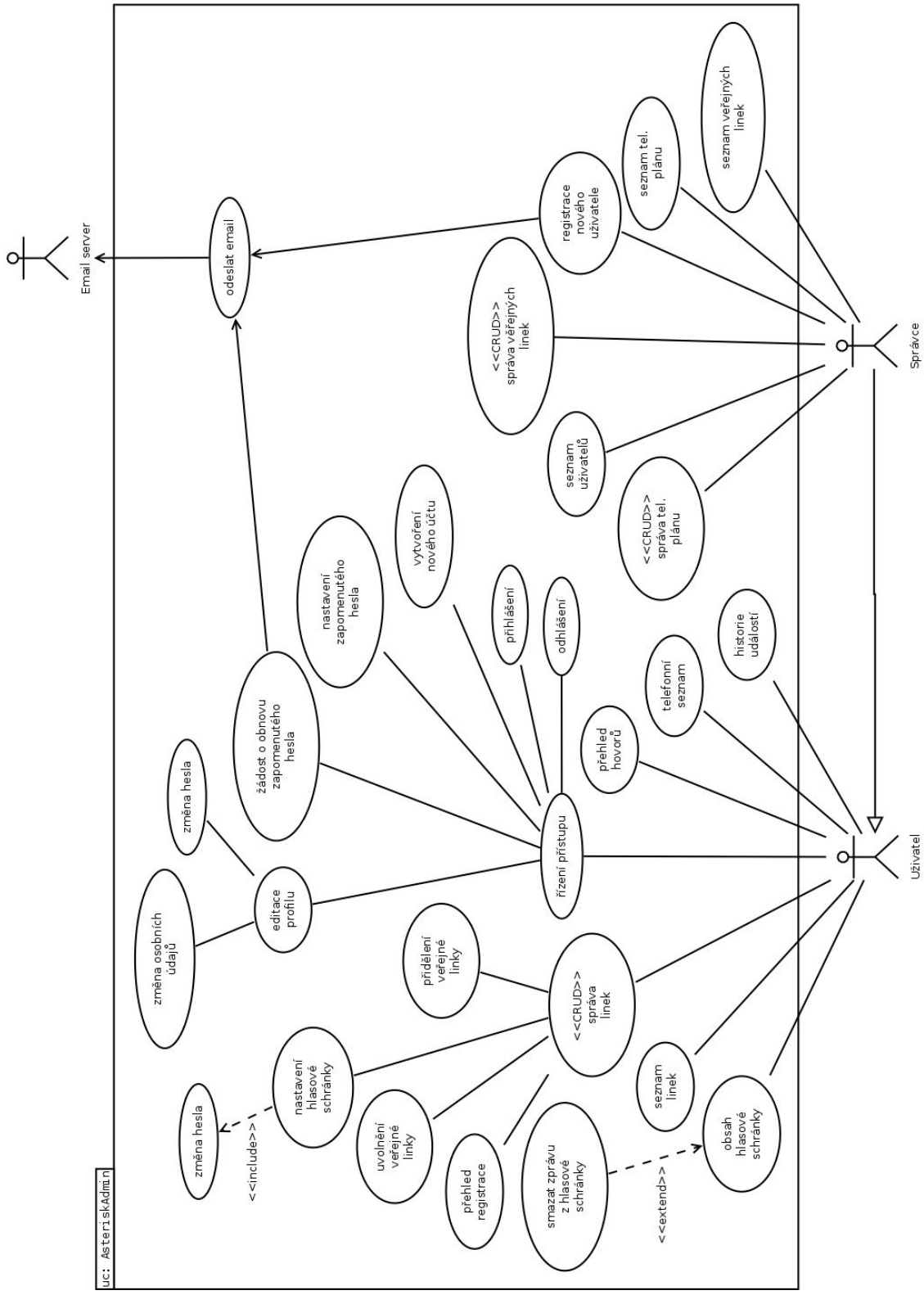
Dostupné z: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29

[20] Druhy, typy a kategorie testů. Testování softwaru [online]. 2011 [cit. 2014-04-25]. Dostupné z: http://testovanisoftwaru.cz/category/druhy-typy-a-kategorie-testu/#unit

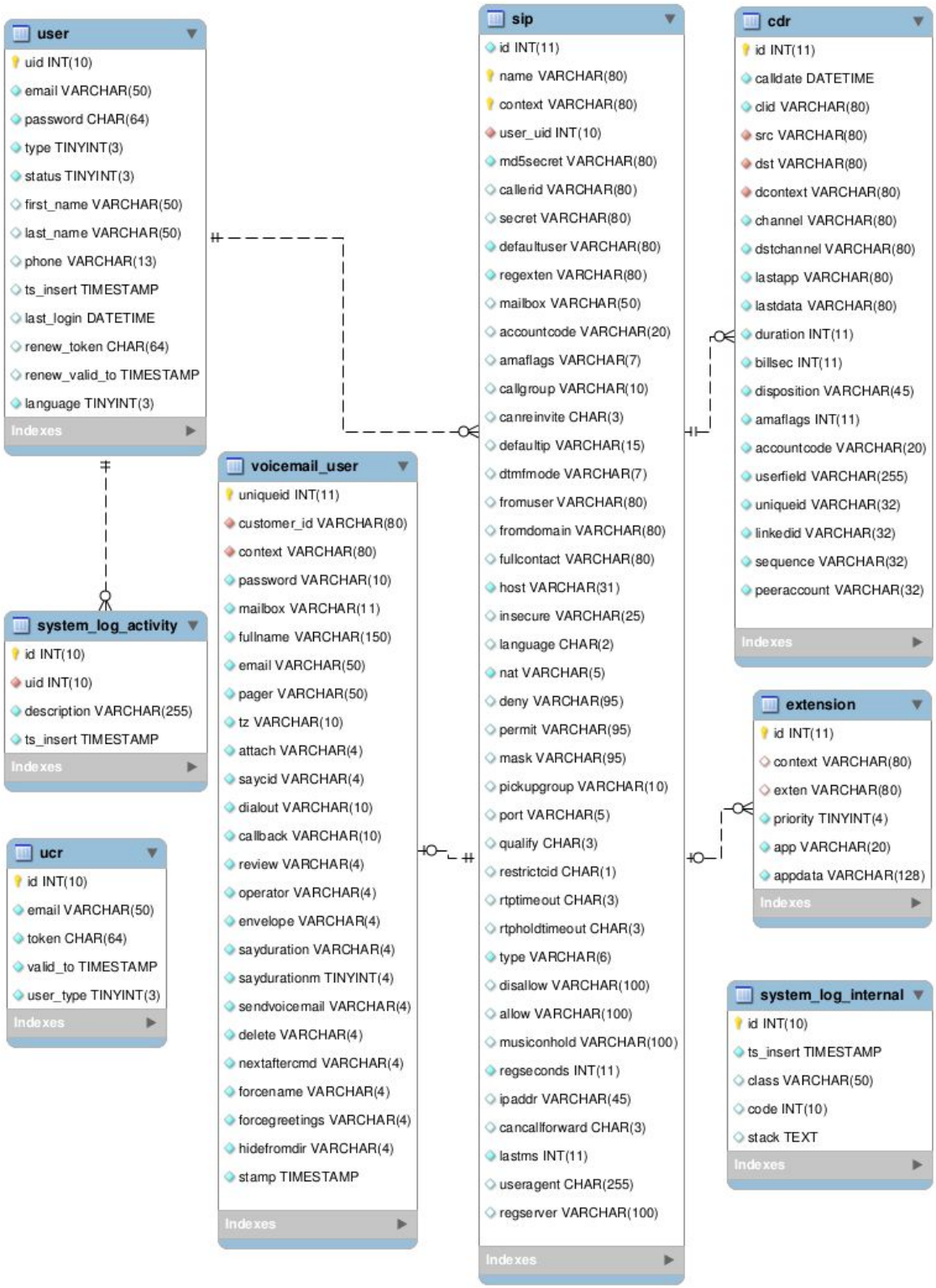
[21] Testování softwaru - funkční a nefunkční testy. Testování softwaru [online]. 2011 [cit. 2014-04-25].

Dostupné z: http://testovanisoftwaru.cz/druhy-typy-a-kategorie-testu/funkcni-a-nefunkcni-testy/

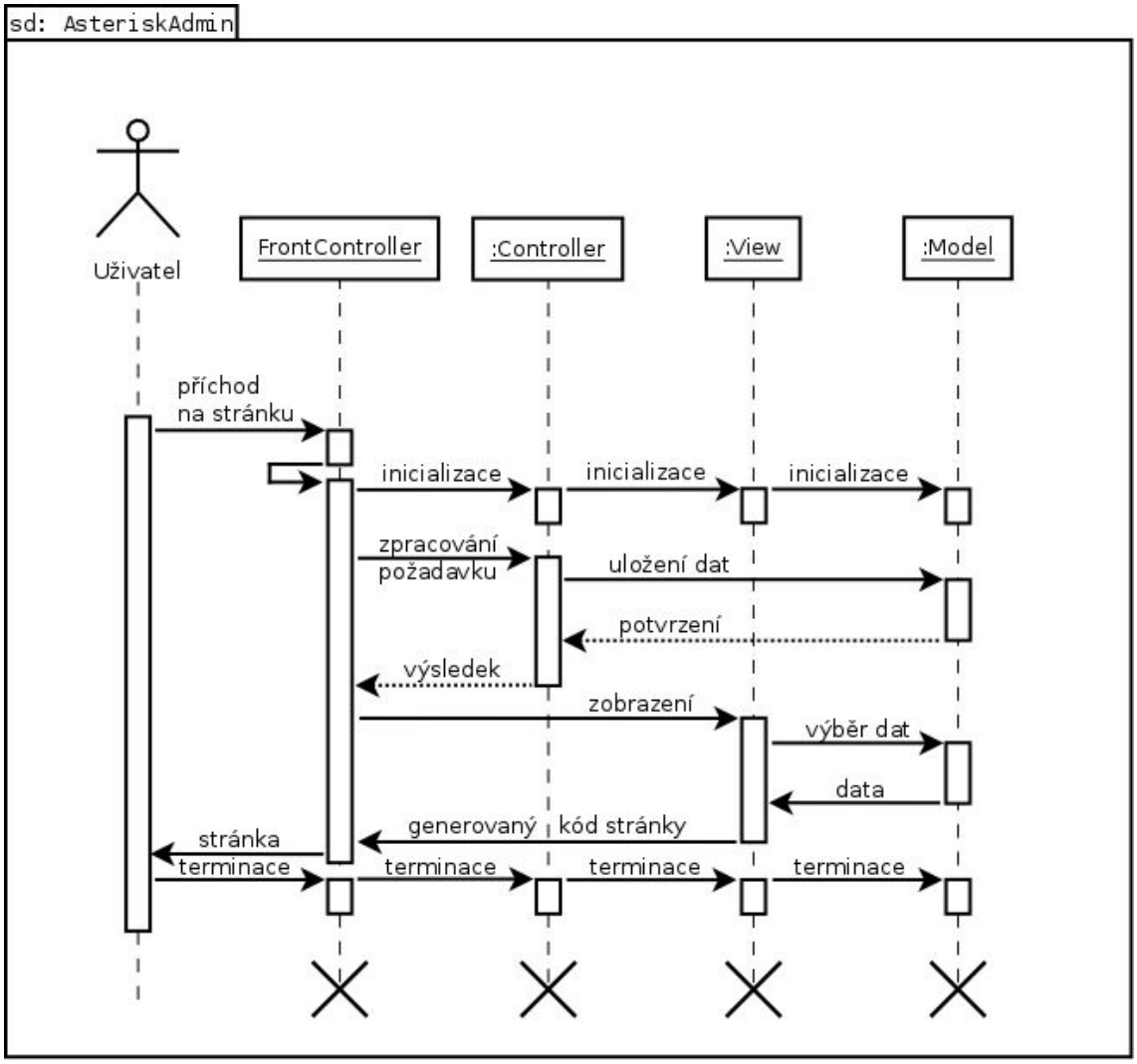
Příloha A – Diagram případů užití



Příloha B – Schéma databáze



Příloha C – Sekvenční diagram



Příloha D – Titulní strana AsteriskAdmin